Introduction

Trajectory is a time dependent description of the path the TCP (Tool Center Point) of an axes group (or axis) moves along motion in 3D space. Additionally to the geometrical description of the space curve, time dependent state variables like velocity, acceleration, jerk, forces etc. are specified.

Click&Move® (C&M) uses standard PLCopen (www.PLCopen.org) and C&M specific Function Blocks (FB) to calculate and execute trajectories.

Note: See also* Multi axis coordinated motion *in the C&M-MC Help File in the C&M Desktop.

FB interfaces at different levels are provided so the user has the option of implementing trajectories in C++ as well. Trajectories can also be implemented by an external user program via the auto generated DLL interface to the IN/OUT ports of C_M_MAIN.sch.

Single-axes Motion Trajectory

The basic building blocks of single axis motion are FBs like **MC_MOVE_ABSOLUTE** and **MC_MOVE_VELOCITY**. Some applications may require a large number of these block instances to program a trajectory.

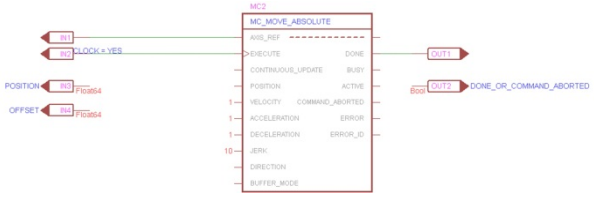

Figure 1 MC_MOVE_ABSOLUTE FB

A C++ User Derived FB (UDFB) may be used to execute single instances of these blocks from a C++ user program.

Note: See the example project* Cpp_FB_Included*. Example project files can be found in the Examples folder in the CandM installation folder. Right click on UDFB/C&M Function Block Files to view C++ code.

Since C&M is essentially a C++ based Visual Programing Language, for periodic application program execution, there is no limit to the complexity of application programs.

FBs can also be executed by an external user program via the auto generated DLL interface to the IN/OUT ports of C_M_MAIN.sch.

However using higher level C&M FBs for coordinated motion like **MC_MOVE_CIRCULAR_ABSOLUTE**, **MC_MOVE_PATH** and **INTERPRET_PATH_MOTION**, programing can greatly be simplified.

Multi-axes Motion Trajectory

The basic building blocks of multi axes coordinated motion application are the **MC_MOVE_LINEAR_ABSOLUTE** and **MC_MOVE_CIRCULAR_ABSOLUTE** FBs.

Figure 2 Coordinated Motion FBs

Note: See CoordinatedMotionBasics example project.

For applications where a large number of these blocks are required a C++ user UDFB may be used to execute instances of **MC_MOVE_LINEAR_ABSOLUTE** and **MC_MOVE_CIRCULAR_ABSOLUTE** FBs.

Multi axes FBs can also be executed by an external user program via the auto generated DLL interface to the IN/OUT ports of C_M_MAIN.sch.

Alternatively, a more convenient solution is to use the **MC_MOVE_PATH** FB.

MC_MOVE_PATH not only executes lines and arcs but also calculates a whole motion path in order to ensure that the Machine starts deceleration in time, from the commanded feed rate to reach a full stop, at zero speed. **MC_MOVE_PATH** is connected to **CM_PATH_SELECT**.

The **CM_PATH_SELECT** FB's **PATH_DESCRIPTION** input is a reference to a te**x**t file **d**ata**b**ase (XDB) containing lines and

arcs and associated profile values (vel, accel, decel and jerk).

A C++ user UDFB can also be used with direct access to this XDB file.

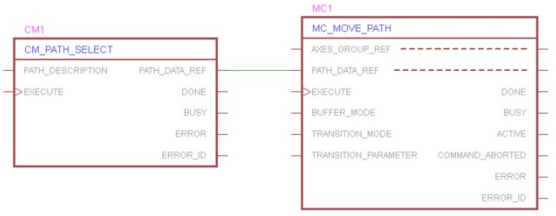

Figure 3 Path Motion

Multi-axes Motion Trajectory with Synchronized Actions

The **INTERPRET_PATH_MOTION** FB executes a trajectory along with synchronized actions like mode change, motion and logic. (E.g. spindle forward/reverse or tool change on a CNC machine.) Any C&M functions (C++or FB) can be used to implement these user defined actions.

INTERPRET_PATH_MOTION is connected to **MI_PATH_SELECT**.

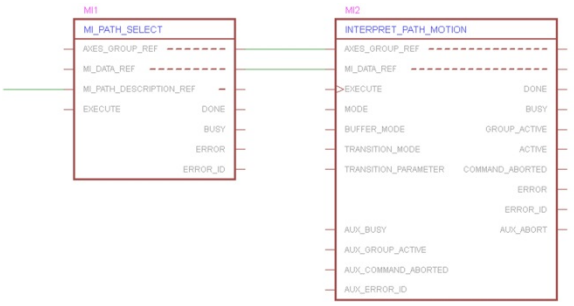

Figure 4 Interpret Path Motion

The **MI_PATH_SELECT** FB's **MI_PATH_DESCRIPTION_REF** input is a reference to a text file database The rows of this XDB file contain, in sequence, lines and arcs (with associated profile values) and also any C&M functions to set operating modes and implement actions. The columns contain variables to specify parameters for the functions.

The **MI_PATH_DESCRIPTION_REF** database text file can be created by the user or generated from industry standard G-code by the **G_MI_PATH_PREPARE** FB.

Embedding G-code into C&M Motion Trajectory

CAD/CAM systems typically have G-code outputs and can be used to auto generate multi-dimensional motion trajectories with synchronized actions.

Alternatively the *C&M motion editor* program can be used to generate the G-code text file.

The user simply keeps adding C&M functions, along with parameters, in sequence.

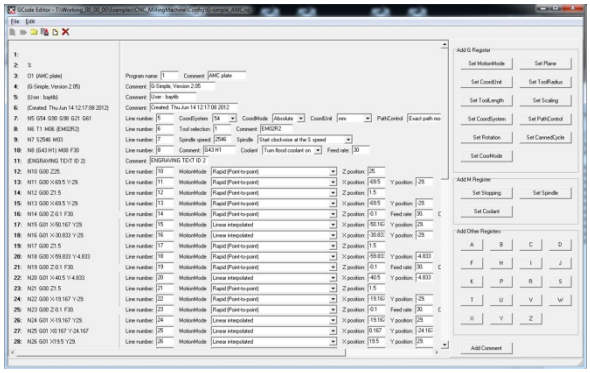

Figure 5 G-Code Editor

The G-code file can be optionally extended with C&M specific registers in order to utilize the full capabilities of C&M functions (E.g. Accel, Decel, Jerk, etc.). This way G-code programing can be seamlessly embedded in the much more powerful language of C&M, which is based on PLCopen and C++.

The register values specified in the G-code file are loaded into a set of C&M variables (the columns of MI_PATH_DESCRIPTION_REF database). For example, the X, Y, Z registers are represented by Float64 while Modal group registers are represented by enumerated data types.

The INTERPRET_PATH_MOTION (and also any lower level FBs) use these variables (or registers) to execute the motion trajectory with associated actions.

Alternatively a C++ user UDFB can be used - accessing the MI_PATH_DESCRIPTION_REF database- to execute the trajectory with associated actions. Of course this C++ UDFB can execute any C&M FBs too.